MaSH Environments

Andrew Rock
School of Information and Communication Technology
Griffith University
Nathan, Queensland, 4111, Australia
a.rock@griffith.edu.au

January 2, 2016

Contents

1 Environment console ........................................ 7
  1.1 Purpose ................................................. 7
  1.2 Rewrites .................................................. 7
  1.3 Console input ............................................ 7
    1.3.1 Purpose ............................................. 7
    1.3.2 Methods ............................................. 7
  1.4 Console output ........................................... 9
    1.4.1 Purpose ............................................. 9
    1.4.2 Methods ............................................. 9
  1.5 Math ...................................................... 10
    1.5.1 Purpose ............................................. 10
    1.5.2 Constants ........................................... 10
    1.5.3 Methods ............................................. 11
  1.6 Strings .................................................. 13
    1.6.1 Purpose ............................................. 13
    1.6.2 Methods ............................................. 13
  1.7 Debugger ............................................... 15
    1.7.1 Purpose ............................................. 15
    1.7.2 Methods ............................................. 16

2 Environment files ........................................... 17
  2.1 Purpose ................................................. 17
  2.2 Rewrites .................................................. 17
  2.3 Console input ............................................ 17
    2.3.1 Purpose ............................................. 17
    2.3.2 Methods ............................................. 17
  2.4 Console output ........................................... 19
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.6.1</td>
<td>Methods</td>
<td>82</td>
</tr>
<tr>
<td>6.7</td>
<td>Output port constants</td>
<td>83</td>
</tr>
<tr>
<td>6.7.1</td>
<td>Constants</td>
<td>83</td>
</tr>
<tr>
<td>6.8</td>
<td>Using motors</td>
<td>83</td>
</tr>
<tr>
<td>6.8.1</td>
<td>Methods</td>
<td>83</td>
</tr>
<tr>
<td>6.9</td>
<td>Using lamps</td>
<td>84</td>
</tr>
<tr>
<td>6.9.1</td>
<td>Methods</td>
<td>84</td>
</tr>
<tr>
<td>6.10</td>
<td>Waiting for fixed times</td>
<td>84</td>
</tr>
<tr>
<td>6.10.1</td>
<td>Methods</td>
<td>84</td>
</tr>
<tr>
<td>6.11</td>
<td>Making sounds</td>
<td>84</td>
</tr>
<tr>
<td>6.11.1</td>
<td>Methods</td>
<td>84</td>
</tr>
<tr>
<td>6.12</td>
<td>Using the LCD</td>
<td>85</td>
</tr>
<tr>
<td>6.12.1</td>
<td>Methods</td>
<td>85</td>
</tr>
<tr>
<td>6.13</td>
<td>Using infra-red communications</td>
<td>86</td>
</tr>
<tr>
<td>6.13.1</td>
<td>Methods</td>
<td>86</td>
</tr>
<tr>
<td>6.14</td>
<td>Math</td>
<td>86</td>
</tr>
<tr>
<td>6.14.1</td>
<td>Purpose</td>
<td>86</td>
</tr>
<tr>
<td>6.14.2</td>
<td>Constants</td>
<td>86</td>
</tr>
<tr>
<td>6.14.3</td>
<td>Methods</td>
<td>86</td>
</tr>
<tr>
<td>6.15</td>
<td>Strings</td>
<td>88</td>
</tr>
<tr>
<td>6.15.1</td>
<td>Purpose</td>
<td>88</td>
</tr>
<tr>
<td>6.15.2</td>
<td>Methods</td>
<td>88</td>
</tr>
<tr>
<td>6.16</td>
<td>Threads</td>
<td>89</td>
</tr>
<tr>
<td>6.16.1</td>
<td>Rewrites</td>
<td>89</td>
</tr>
<tr>
<td>6.16.2</td>
<td>Methods</td>
<td>89</td>
</tr>
</tbody>
</table>
1 Environment console

1.1 Purpose

This environment supports programming for a console program that reads from standard input and writes to standard output.

1.2 Rewrites

*mandatory*

```java
void main ()
```

*Purpose:* A program that is organised into methods must have a `main` method (a procedure with no arguments). This will be the first method to execute. *mashc* automatically rewrites this method to conform to standard Java.

1.3 Console input

1.3.1 Purpose

The following are methods for reading from standard input (which is usually the keyboard) and for checking to see if there is more input to read.

1.3.2 Methods

```java
int readInt ()
```

*Purpose:* Returns the next integer from standard input.

*Precondition:* Will cause a run time error unless there is an integer in standard input to read. Use `isNextInt()` to check first.

```java
long readLong ()
```

*Purpose:* Returns the next long from standard input.

*Precondition:* Will cause a run time error unless there is a long in standard input to read. Use `isNextLong()` to check first.

```java
boolean readBoolean ()
```

*Purpose:* Returns the next boolean from standard input.

*Precondition:* Will cause a run time error unless there is a boolean in standard input to read. Use `isNextBoolean()` to check first.

```java
double readDouble ()
```

Purpose: Returns the next double from standard input.

Precondition: Will cause a run time error unless there is a double in standard input to read. Use isNextDouble() to check first.

float readFloat ()

Purpose: Returns the next float from standard input.

Precondition: Will cause a run time error unless there is a float in standard input to read. Use isNextFloat() to check first.

String readWord ()

Purpose: Returns the next word as a String. A “word” is a sequence of one-or-more non-whitespace characters.

Precondition: Will cause a run time error unless there is a word in standard input to read. Use isNextWord() to check first.

String readLine ()

Purpose: Returns the next line of text as a String. A line is a sequence of zero-or-more characters terminated by the end of line, which is not returned as part of the line.

Precondition: Will cause a run time error unless there is a line in standard input to read. Use isNextLine() to check first.

boolean isNextInt ()

Purpose: Returns true if and only if there is an integer in standard input available to read, that is readInt() would succeed.

boolean isNextLong ()

Purpose: Returns true if and only if there is a long in standard input available to read, that is readLong() would succeed.

boolean isNextBoolean ()

Purpose: Returns true if and only if there is an boolean in standard input available to read, that is readBoolean() would succeed.

boolean isNextDouble ()

Purpose: Returns true if and only if there is a double in standard input available to read, that is readDouble() would succeed.
boolean isNextFloat ()

*Purpose*: Returns `true` if and only if there is a float in standard input available to read, that is `readFloat()` would succeed.

boolean isNextWord ()

*Purpose*: Returns `true` if and only if there is a word in standard input available to read, that is `readWord()` would succeed.

boolean isNextLine ()

*Purpose*: Returns `true` if and only if there is a line in standard input available to read, that is `readLine()` would succeed.

### 1.4 Console output

#### 1.4.1 Purpose

The following are methods for printing to standard output (which is usually the terminal window).

#### 1.4.2 Methods

void print (char c)

*Purpose*: Writes character `c` to standard output.

void print (String s)

*Purpose*: Writes string `s` to standard output.

void print (long i)

*Purpose*: Writes integral `i` to standard output.

void print (boolean p)

*Purpose*: Writes boolean `p` to standard output.

void print (double x)

*Purpose*: Writes floating point number `x` to standard output.

void println (char c)

*Purpose*: Writes character `c` and then a newline to standard output.

void println ()
Purpose: Writes a newline to standard output.

void println (String s)

Purpose: Writes string s and then a newline to standard output.

void println (long i)

Purpose: Writes integral i and then a newline to standard output.

void println (boolean p)

Purpose: Writes boolean p and then a newline to standard output.

void println (double x)

Purpose: Writes floating point number x and then a newline to standard output.

1.5 Math

1.5.1 Purpose

The following are some commonly used numeric constants and functions.

1.5.2 Constants

final int MAX_INT

Purpose: A constant holding the maximum value an int can have, $2^{31} - 1$.

final int MIN_INT

Purpose: A constant holding the minimum value an int can have, $-2^{31}$.

final long MAX_LONG

Purpose: A constant holding the maximum value a long can have, $2^{63} - 1$.

final long MIN_LONG

Purpose: A constant holding the minimum value a long can have, $-2^{63}$.

final double PI

Purpose: The closest double approximation to $\pi$.

final double E

Purpose: The closest double approximation to $e$. 
1.5.3 Methods

double abs (double a)

    Purpose: Returns the absolute value of a.

float abs (float a)

    Purpose: Returns the absolute value of a.

long abs (long a)

    Purpose: Returns the absolute value of a.

int abs (int a)

    Purpose: Returns the absolute value of a.

double ceil (double a)

    Purpose: Returns the least double value that is greater than or
               equal to a and equal to an integer.

double exp (double x)

    Purpose: Returns $e^x$, that is Euler’s constant $e$ raised to power $x$.

double floor (double a)

    Purpose: Returns the greatest double value that is less than or
               equal to a and equal to an integer.

double log (double x)

    Purpose: Returns the natural logarithm of $x$.

double rint (double x)

    Purpose: Returns the closest mathematical integer to $x$.

long round (double a)

    Purpose: Returns the closest long to a.

int round (float a)

    Purpose: Returns the closest int to a.

double sqrt (double a)

    Purpose: Returns the square root of a.
Precondition: \( a \geq 0.0 \).

double pow (double a, double b)

Purpose: Returns \( a \) raised to the power \( b \), \( a^b \).

double sin (double a)

Purpose: Returns the trigonometric sine of \( a \) radians.

double cos (double a)

Purpose: Returns the trigonometric cosine of \( a \) radians.

double tan (double a)

Purpose: Returns the trigonometric tangent of \( a \) radians.

double asin (double a)

Purpose: Returns the trigonometric arc sine of \( a \) in radians.

double acos (double a)

Purpose: Returns the trigonometric arc cosine of \( a \) in radians.

double atan (double a)

Purpose: Returns the trigonometric arc tangent of \( a \) in radians.

double atan2 (double y, double x)

Purpose: Returns the angle \( \theta \) from the conversion of rectangular coordinates \((x, y)\) to polar coordinates \((r, \theta)\).

double max (double a, double b)

Purpose: Returns the greater of \( a \) and \( b \).

float max (float a, float b)

Purpose: Returns the greater of \( a \) and \( b \).

int max (int a, int b)

Purpose: Returns the greater of \( a \) and \( b \).

long max (long a, long b)

Purpose: Returns the greater of \( a \) and \( b \).
double min (double a, double b)

Purpose: Returns the lesser of a and b.

float min (float a, float b)

Purpose: Returns the lesser of a and b.

int min (int a, int b)

Purpose: Returns the lesser of a and b.

long min (long a, long b)

Purpose: Returns the lesser of a and b.

double random ()

Purpose: Returns a random value \( x \) such that \( 0.0 \leq x < 1.0 \).

1.6 Strings

1.6.1 Purpose
The following are methods for working with Strings.

1.6.2 Methods

int length (String s)

Purpose: Returns the length of s.

char charAt (String s, int i)

Purpose: Returns the character at position \( i \) in s.

Precondition: \( 0 \leq i < \text{length}(s) \).

boolean equals (String a, String b)

Purpose: Returns \text{true} if and only if a contains the same sequence of characters as in b.

String format (boolean p, int w)

Purpose: Returns \text{p} converted to a string, padded with spaces to a minimum width \(|w|\). If \( w \) is negative, the result is left-justified, otherwise right-justified.

Examples:
format(true, 10) returns "\text{true}\""; and
format(true, -10) returns "\text{true}\"."
String format (char c, int w)

Purpose: Returns c converted to a string, padded with spaces to a minimum width |w|. If w is negative, the result is left-justified, otherwise right-justified.

Examples:
format('a', 3) returns " a"; and
format('a', -3) returns "a  ".

String format (double d, int w, char f, int p)

Purpose: Returns d converted to a string, padded with spaces to a minimum width |w|. If w is negative, the result is left-justified, otherwise right-justified. f controls the format: 'e' selects scientific notation; 'f' selects fixed point; or 'g' selects the best format depending on the number and p. For 'e' and 'f', p is the number of decimal digits to display after the decimal point, but for 'g' it is the total number of digits.

Examples:
format(1234.56789, 12, 'e', 4) returns " 1.2346e+03";
format(1234.56789, 12, 'f', 4) returns " 1234.5679";
format(1234.56789, 12, 'g', 4) returns " 1235";
format(0.000001234567, 12, 'e', 4) returns " 1.2346e-06";
format(0.000001234567, 12, 'f', 4) returns " 0.0000";
and
format(0.000001234567, 12, 'g', 4) returns " 1.235e-06".

String format (long l, int w)

Purpose: Returns l converted to a string, padded with spaces to a minimum width |w|. If w is negative, the result is left-justified, otherwise right-justified.

Examples:
format(42, 5) returns "  42"; and
format(42, -5) returns "42   ".

String format (String s, int w)

Purpose: Returns s padded with spaces to a minimum width |w|. If w is negative, the result is left-justified, otherwise right-justified.

Examples:
format("aaa", 5) returns "   aaa"; and
format("aaa", -5) returns "aaa   ".

boolean parseBoolean (String s)
Purpose: Returns s converted to a boolean.

```java
int parseInt (String s)
```

Purpose: Returns s converted to an int.

```java
long parseLong (String s)
```

Purpose: Returns s converted to a long.

```java
float parseFloat (String s)
```

Purpose: Returns s converted to a float.

```java
double parseDouble (String s)
```

Purpose: Returns s converted to a double.

### 1.7 Debugger

#### 1.7.1 Purpose

This environment supports a debugger with a graphical user interface. It allows the user to slow down the execution of a program, so that the flow of control through the program may be observed and the values in variables monitored.

To activate the debugger, compile your program with the `mashc +debug` option.

While the program is running under the control of the debugger, the original MaSH program’s source code is displayed on the left, with a pointer showing the statement that is about to execute. Sometimes it will point to a closing brace, indicating that the current method is about to be exited, or a loop guard is about to be tested again, depending on the context.

The contents of all the program's variables are displayed on the right in a table. The entries in the scope column, consist of a number and a word. The number indicates which method invocation the variable belongs to. For non-method MaSH programs the number is always 0. For method MaSH programs, 0 indicates a global variable, 1 indicates the main method, 2 indicates a method called from main, 3 indicates a method called from 2, etc. The word can be: `global`; `local`; or `param`. Variables are added to the table as they come into scope and removed again when their enclosing block exits. Anything changed on the last program step is displayed in green.

Limitations: The debugger can not display values of arrays with 3 or more dimensions. The debugger can not step through methods that run in the program’s event dispatch thread (for example, the `paintWindow` method in the `graphics` environment). This GUI debugger can not work in environments without a big screen (for example, Lego Mindstorms).
1.7.2 Methods

void suspendDebugger ()

Purpose: Suspend the debugger.
Calling this in your MaSH program disables the debugger from pausing your program or updating the debugger window.
If the debugger has not been included at compile time, this procedure does nothing at all.

void resumeDebugger ()

Purpose: Resume the debugger.
Calling this in your MaSH program re-enables the debugger to pause your program and update the debugger window.
If the debugger has not been included at compile time, this procedure does nothing at all.
Do not call this in methods that run in the program’s event dispatch thread (for example, the paintWindow method in the graphics environment).
2 Environment files

2.1 Purpose

This environment supports programming for a console program that reads from standard input and writes to standard output, as per environment console, but also adds the capability to read and write files on disk.

2.2 Rewrites

*mandatory*

```java
void main ()
```

*Purpose:* A program that is organised into methods must have a `main` method (a procedure with no arguments). This will be the first method to execute. *mashc* automatically rewrites this method to conform to standard Java.

2.3 Console input

2.3.1 Purpose

The following are methods for reading from standard input (which is usually the keyboard) and for checking to see if there is more input to read.

2.3.2 Methods

```java
int readInt ()
```

*Purpose:* Returns the next integer from standard input.

*Precondition:* Will cause a run time error unless there is an integer in standard input to read. Use `isNextInt()` to check first.

```java
long readLong ()
```

*Purpose:* Returns the next long from standard input.

*Precondition:* Will cause a run time error unless there is a long in standard input to read. Use `isNextLong()` to check first.

```java
boolean readBoolean ()
```

*Purpose:* Returns the next boolean from standard input.

*Precondition:* Will cause a run time error unless there is a boolean in standard input to read. Use `isNextBoolean()` to check first.

```java
double readDouble ()
```
Purpose: Returns the next double from standard input.

Precondition: Will cause a run time error unless there is a double in standard input to read. Use isNextDouble() to check first.

float readFloat ()

Purpose: Returns the next float from standard input.

Precondition: Will cause a run time error unless there is a float in standard input to read. Use isNextFloat() to check first.

String readWord ()

Purpose: Returns the next word as a String. A “word” is a sequence of one-or-more non-whitespace characters.

Precondition: Will cause a run time error unless there is a word in standard input to read. Use isNextWord() to check first.

String readLine ()

Purpose: Returns the next line of text as a String. A line is a sequence of zero-or-more characters terminated by the end of line, which is not returned as part of the line.

Precondition: Will cause a run time error unless there is a line in standard input to read. Use isNextLine() to check first.

boolean isNextInt ()

Purpose: Returns true if and only if there is an integer in standard input available to read, that is readInt() would succeed.

boolean isNextLong ()

Purpose: Returns true if and only if there is a long in standard input available to read, that is readLong() would succeed.

boolean isNextBoolean ()

Purpose: Returns true if and only if there is an boolean in standard input available to read, that is readBoolean() would succeed.

boolean isNextDouble ()

Purpose: Returns true if and only if there is a double in standard input available to read, that is readDouble() would succeed.
boolean isNextFloat ()

    *Purpose:* Returns `true` if and only if there is a float in standard input available to read, that is `readFloat()` would succeed.

boolean isNextWord ()

    *Purpose:* Returns `true` if and only if there is a word in standard input available to read, that is `readWord()` would succeed.

boolean isNextLine ()

    *Purpose:* Returns `true` if and only if there is a line in standard input available to read, that is `readLine()` would succeed.

2.4 Console output

2.4.1 Purpose

The following are methods for printing to standard output (which is usually the terminal window).

2.4.2 Methods

void print (char c)

    *Purpose:* Writes character `c` to standard output.

void print (String s)

    *Purpose:* Writes string `s` to standard output.

void print (long i)

    *Purpose:* Writes integral `i` to standard output.

void print (boolean p)

    *Purpose:* Writes boolean `p` to standard output.

void print (double x)

    *Purpose:* Writes floating point number `x` to standard output.

void println (char c)

    *Purpose:* Writes character `c` and then a newline to standard output.

void println ()
### Purpose

```java
void println (String s)
```

*Purpose:* Writes string `s` and then a newline to standard output.

```java
void println (long i)
```

*Purpose:* Writes integral `i` and then a newline to standard output.

```java
void println (boolean p)
```

*Purpose:* Writes boolean `p` and then a newline to standard output.

```java
void println (double x)
```

*Purpose:* Writes floating point number `x` and then a newline to standard output.

#### 2.5 Opening and closing files

##### 2.5.1 Purpose

These are the methods by which we open and close files.

When a file is opened, the method that opens the file will return an `int`, a fileID, which identifies that file to the methods for reading or writing. If the returned `int` is negative, the file could not be opened for some reason.

Files opened for reading can not be written to. Files opened for writing can not be read from.

Files opened for reading or writing must be closed by calling the `close` method before your program ends.

Files are opened by specifying the `path` to it. Example `paths`: `fred.txt`; `E:\holiday\itenerary.txt`.

##### 2.5.2 Methods

```java
int openRead (String path)
```

*Purpose:* `openRead(path)` opens the file at `path` for reading and returns the unique ID by which this file may be accessed. IDs are always non-negative. If a negative number is returned, the file could not be opened.

*Precondition:* There must be a file at `path` to read from.

```java
int openWrite (String path)
```
**Purpose:** `openWrite(path)` opens the file at `path` for writing and returns the unique ID by which this file may be accessed. IDs are always non-negative. If a negative number is returned, the file could not be opened. If there is an existing file at `path`, its contents are discarded.

```java
void close (int fileID)
```

**Purpose:** `close(fileID)` closes the file selected by `fileID`.

### 2.6 Reading from files

#### 2.6.1 Purpose

The following are methods for reading from files and for checking to see if there is more input to read from them.

#### 2.6.2 Methods

```java
int readInt (int fileID)
```

**Purpose:** Returns the next integer from the file selected by `fileID`.

**Precondition:** `fileID` selects a file that is open for reading.

**Precondition:** Will cause a run time error unless there is an integer in the file to read. Use `isNextInt(fileID)` to check first.

```java
long readLong (int fileID)
```

**Purpose:** Returns the next long from the file selected by `fileID`.

**Precondition:** `fileID` selects a file that is open for reading.

**Precondition:** Will cause a run time error unless there is a long in the file to read. Use `isNextLong(fileID)` to check first.

```java
boolean readBoolean (int fileID)
```

**Purpose:** Returns the next boolean from the file selected by `fileID`.

**Precondition:** `fileID` selects a file that is open for reading.

**Precondition:** Will cause a run time error unless there is a boolean in the file to read. Use `isNextBoolean(fileID)` to check first.

```java
double readDouble (int fileID)
```

**Purpose:** Returns the next double from the file selected by `fileID`.

21
Precondition: fileID selects a file that is open for reading.

Precondition: Will cause a run time error unless there is a double in the file to read. Use isNextDouble(fileID) to check first.

float readFloat (int fileID)

Purpose: Returns the next float from the file selected by fileID.

Precondition: fileID selects a file that is open for reading.

Precondition: Will cause a run time error unless there is a float in the file to read. Use isNextFloat(fileID) to check first.

String readWord (int fileID)

Purpose: Returns the next word as a String from the file selected by fileID. A “word” is a sequence of one-or-more non-whitespace characters.

Precondition: fileID selects a file that is open for reading.

Precondition: Will cause a run time error unless there is a word in the file to read. Use isNextWord(fileID) to check first.

String readLine (int fileID)

Purpose: Returns the next line of text as a String from the file selected by fileID. A line is a sequence of zero-or-more characters terminated by the end of line, which is not returned as part of the line.

Precondition: fileID selects a file that is open for reading.

Precondition: Will cause a run time error unless there is a line in the file to read. Use isNextLine(fileID) to check first.

boolean isNextInt (int fileID)

Purpose: Returns true if and only if there is an integer in the file selected by fileID available to read, that is readInt(fileID) would succeed.

Precondition: fileID selects a file that is open for reading.

boolean isNextLong (int fileID)

Purpose: Returns true if and only if there is a long in the file selected by fileID available to read, that is readLong(fileID) would succeed.
Precondition: fileId selects a file that is open for reading.

boolean isNextBoolean (int fileId)

Purpose: Returns true if and only if there is an boolean in the file available to read, that is readBoolean(fileId) would succeed.

boolean isNextDouble (int fileId)

Purpose: Returns true if and only if there is a double in the file selected by fileId available to read, that is readDouble(fileId) would succeed.

Precondition: fileId selects a file that is open for reading.

boolean isNextFloat (int fileId)

Purpose: Returns true if and only if there is a float in the file selected by fileId available to read, that is readFloat(fileId) would succeed.

Precondition: fileId selects a file that is open for reading.

boolean isNextWord (int fileId)

Purpose: Returns true if and only if there is a word in the file selected by fileId available to read, that is readWord(fileId) would succeed.

boolean isNextLine (int fileId)

Purpose: Returns true if and only if there is a line in the file selected by fileId available to read, that is readLine(fileId) would succeed.

Precondition: fileId selects a file that is open for reading.

2.7 Writing to files

2.7.1 Purpose

The following are methods for printing to files.
2.7.2 Methods

void print (int fileID, char c)

Purpose: Writes character c to the file selected by fileID.

Precondition: fileID selects a file that is open for writing.

void print (int fileID, String s)

Purpose: Writes string s to the file selected by fileID.

Precondition: fileID selects a file that is open for writing.

void print (int fileID, long i)

Purpose: Writes integral i to the file selected by fileID.

Precondition: fileID selects a file that is open for writing.

void print (int fileID, boolean p)

Purpose: Writes boolean p to the file selected by fileID.

Precondition: fileID selects a file that is open for writing.

void print (int fileID, double x)

Purpose: Writes floating point number x to the file selected by fileID.

Precondition: fileID selects a file that is open for writing.

void println (int fileID, char c)

Purpose: Writes character c and then a newline to the file selected by fileID.

Precondition: fileID selects a file that is open for writing.

void println (int fileID, String s)

Purpose: Writes string s and then a newline to the file selected by fileID.

Precondition: fileID selects a file that is open for writing.

void println (int fileID, long i)

Purpose: Writes integral i and then a newline to the file selected by fileID.
Precondition: fileID selects a file that is open for writing.

```java
void println (int fileID, boolean p)
```

Purpose: Writes boolean p and then a newline to the file selected by fileID.

Precondition: fileID selects a file that is open for writing.

```java
void println (int fileID, double x)
```

Purpose: Writes floating point number x and then a newline to the file selected by fileID.

Precondition: fileID selects a file that is open for writing.

### 2.8 Command line arguments

#### 2.8.1 Purpose

These methods give access to the command line arguments.

#### 2.8.2 Methods

```java
String[] args ()
```

Purpose: Returns the array of command line arguments.

```java
int numArgs ()
```

Purpose: Returns the number of command line arguments.

### 2.9 Math

#### 2.9.1 Purpose

The following are some commonly used numeric constants and functions.

#### 2.9.2 Constants

```java
final int MAX_INT
```

Purpose: A constant holding the maximum value an int can have, $2^{31} - 1$.

```java
final int MIN_INT
```

Purpose: A constant holding the minimum value an int can have, $-2^{31}$.
final long MAX_LONG

*Purpose:* A constant holding the maximum value a *long* can have, $2^{63} - 1$.

final long MIN_LONG

*Purpose:* A constant holding the minimum value a *long* can have, $-2^{63}$.

final double PI

*Purpose:* The closest *double* approximation to $\pi$.

final double E

*Purpose:* The closest *double* approximation to $e$.

**2.9.3 Methods**

double abs (double a)

*Purpose:* Returns the absolute value of $a$.

float abs (float a)

*Purpose:* Returns the absolute value of $a$.

long abs (long a)

*Purpose:* Returns the absolute value of $a$.

int abs (int a)

*Purpose:* Returns the absolute value of $a$.

double ceil (double a)

*Purpose:* Returns the least double value that is greater than or equal to $a$ and equal to an integer.

double exp (double x)

*Purpose:* Returns $e^x$, that is Euler’s constant $e$ raised to power $x$.

double floor (double a)

*Purpose:* Returns the greatest double value that is less than or equal to $a$ and equal to an integer.
double log (double x)

*Purpose*: Returns the natural logarithm of \( x \).

double rint (double x)

*Purpose*: Returns the closest mathematical integer to \( x \).

long round (double a)

*Purpose*: Returns the closest \( \text{long} \) to \( a \).

int round (float a)

*Purpose*: Returns the closest \( \text{int} \) to \( a \).

double sqrt (double a)

*Purpose*: Returns the square root of \( a \).

*Precondition*: \( a \geq 0.0 \).

double pow (double a, double b)

*Purpose*: Returns \( a \) raised to the power \( b \), \( a^b \).

double sin (double a)

*Purpose*: Returns the trigonometric sine of \( a \) radians.

double cos (double a)

*Purpose*: Returns the trigonometric cosine of \( a \) radians.

double tan (double a)

*Purpose*: Returns the trigonometric tangent of \( a \) radians.

double asin (double a)

*Purpose*: Returns the trigonometric arc sine of \( a \) in radians.

double acos (double a)

*Purpose*: Returns the trigonometric arc cosine of \( a \) in radians.

double atan (double a)

*Purpose*: Returns the trigonometric arc tangent of \( a \) in radians.

double atan2 (double y, double x)
Purpose: Returns the angle \( \theta \) from the conversion of rectangular coordinates \((x, y)\) to polar coordinates \((r, \theta)\).

double max (double a, double b)

Purpose: Returns the greater of \(a\) and \(b\).

float max (float a, float b)

Purpose: Returns the greater of \(a\) and \(b\).

int max (int a, int b)

Purpose: Returns the greater of \(a\) and \(b\).

long max (long a, long b)

Purpose: Returns the greater of \(a\) and \(b\).

double min (double a, double b)

Purpose: Returns the lesser of \(a\) and \(b\).

float min (float a, float b)

Purpose: Returns the lesser of \(a\) and \(b\).

int min (int a, int b)

Purpose: Returns the lesser of \(a\) and \(b\).

long min (long a, long b)

Purpose: Returns the lesser of \(a\) and \(b\).

double random ()

Purpose: Returns a random value \(x\) such that \(0.0 \leq x < 1.0\).

2.10 Strings

2.10.1 Purpose

The following are methods for working with Strings.
2.10.2 Methods

int length (String s)

Purpose: Returns the length of s.

c char charAt (String s, int i)

Purpose: Returns the character at position i in s.

Precondition: 0 ≤ i < length(s).

boolean equals (String a, String b)

Purpose: Returns true if and only if a contains the same sequence of characters as in b.

String format (boolean p, int w)

Purpose: Returns p converted to a string, padded with spaces to a minimum width |w|. If w is negative, the result is left-justified, otherwise right-justified.

Examples:
format(true, 10) returns " true"; and
format(true, -10) returns "true   ".

String format (char c, int w)

Purpose: Returns c converted to a string, padded with spaces to a minimum width |w|. If w is negative, the result is left-justified, otherwise right-justified.

Examples:
format('a', 3) returns " a"; and
format('a', -3) returns "a   ".

String format (double d, int w, char f, int p)

Purpose: Returns d converted to a string, padded with spaces to a minimum width |w|. If w is negative, the result is left-justified, otherwise right-justified. f controls the format: ’e’ selects scientific notation; ’f’ selects fixed point; or ’g’ selects the best format depending on the number and p. For ’e’ and ’f’, p is the number of decimal digits to display after the decimal point, but for ’g’ it is the total number of digits.

Examples:
format(1234.56789, 12, ’e’, 4) returns " 1.2346e+03";
format(1234.56789, 12, ’f’, 4) returns " 1234.5679";
format(1234.56789, 12, 'g', 4) returns "1235";
format(0.000001234567, 12, 'e', 4) returns "1.2346e-06";
format(0.000001234567, 12, 'f', 4) returns "0.0000";
and
format(0.000001234567, 12, 'g', 4) returns "1.235e-06".

String format (long l, int w)

Purpose: Returns l converted to a string, padded with spaces to a minimum width |w|. If w is negative, the result is left-justified, otherwise right-justified.

Examples:
format(42, 5) returns "42"; and
format(42, -5) returns "42  ".

String format (String s, int w)

Purpose: Returns s padded with spaces to a minimum width |w|. If w is negative, the result is left-justified, otherwise right-justified.

Examples:
format("aaa", 5) returns "aaa"; and
format("aaa", -5) returns "aaa  ".

boolean parseBoolean (String s)

Purpose: Returns s converted to a boolean.

int parseInt (String s)

Purpose: Returns s converted to an int.

long parseLong (String s)

Purpose: Returns s converted to a long.

float parseFloat (String s)

Purpose: Returns s converted to a float.

double parseDouble (String s)

Purpose: Returns s converted to a double.
2.11 Debugger

2.11.1 Purpose

This environment supports a debugger with a graphical user interface. It allows the user to slow down the execution of a program, so that the flow of control through the program may be observed and the values in variables monitored.

To activate the debugger, compile your program with the `mashc +debug` option.

While the program is running under the control of the debugger, the original MaSH program’s source code is displayed on the left, with a pointer showing the statement that is about to execute. Sometimes it will point to a closing brace, indicating that the current method is about to be exited, or a loop guard is about to be tested again, depending on the context.

The contents of all the program’s variables are displayed on the right in a table. The entries in the `scope` column, consist of a number and a word. The number indicates which method invocation the variable belongs to. For no-method MaSH programs the number is always 0. For method MaSH programs, 0 indicates a global variable, 1 indicates the `main` method, 2 indicates a method called from `main`, 3 indicates a method called from 2, etc. The word is blank for variables in a no-method program, unless the variable is declared in a block, which makes it `local`. In method programs, the word can be: `global; local; or param`. Variables are added to the table as they come into scope and removed again when their enclosing block exits. Anything changed on the last program step is displayed in green.

Limitations: The debugger can not display values of arrays with 3 or more dimensions. The debugger can not step through methods that run in the program’s event dispatch thread (for example, the `paintWindow` method in the `graphics` environment). This GUI debugger can not work in environments without a big screen (for example, Lego Mindstorms).

2.11.2 Methods

```java
void suspendDebugger ()

Purpose: Suspend the debugger.

Calling this in your MaSH program disables the debugger from pausing your program or updating the debugger window.

If the debugger has not been incuded at compile time, this proce- dure does nothing at all.

void resumeDebugger ()

Purpose: Resume the debugger.

Calling this in your MaSH program re-enables the debugger to pause your program and update the debugger window.
```
If the debugger has not been included at compile time, this procedure does nothing at all.

Do not call this in methods that run in the program’s event dispatch thread (for example, the `paintWindow` method in the `graphics` environment).
3 Environment graphics

3.1 Purpose

This environment supports drawing in a window, and responding to mouse and keyboard events. It also supports console input and output.

3.2 Rewrites

\textit{mandatory}

\texttt{void main ()}

\textit{Purpose}: A graphics program must have a \texttt{main} method (a procedure with no arguments). This will be the first method to execute. \texttt{mashc} automatically rewrites this method to conform to standard Java.

In the \texttt{main} method, a program must at least call \texttt{setFrameVisible(true)} to make the window visible.

The \texttt{main} method, and any method called by \texttt{main} may not call any of the drawing methods provided by this environment.

The \texttt{main} method may call \texttt{animate(int)} just once to start the automatic redrawing of the window for animation.

\textit{mandatory}

\texttt{void paintWindow ()}

\textit{Purpose}: This is the method called to draw the graphics. Every graphics program must define this procedure. The drawing methods provided by this environment may only be called from this method or from a method called from this method.

Do not call this method directly from \texttt{main}. It will be called when the window is made visible and then any time the window needs to be redrawn.

3.3 Setting up the window

3.3.1 Purpose

Use these methods to set up the window into which all drawing is done.

In Java, the window is known as a \texttt{Frame}.

3.3.2 Methods

\texttt{void setFrameSize (int width, int height)}

\textit{Purpose}: Sets the size of the window with \texttt{width} and \texttt{height}, both in pixels. Call this before making the window visible.
void setFrameTitle (String title)

    Purpose: Set the title of the window. Call this before making the window visible.

void setFrameVisible (boolean visibility)

    Purpose: Set the visibility of the window. true means visible.

3.4 Content area dimensions

3.4.1 Purpose

The size of the window, set with setFrameSize(int, int), is the size of the whole window including its title bar and resizing control areas. The actual size of the content area into which we can draw is platform dependent, and the as the user resizes the window may change. These methods allow the program to find out the true current dimensions of the content area.

3.4.2 Methods

int contentWidth ()

    Purpose: This returns the current width of the content area of the frame in pixels.

int contentHeight ()

    Purpose: This returns the current height of the content area of the frame in pixels.

3.5 Drawing

3.5.1 Purpose

Use these methods draw in the window.

These methods may only be called by the paintWindow() method or by any method called by paintWindow().

All lengths are in pixels. All angles are in degrees.

All rectangles are specified by (x, y, width, height), where (x,y) are the coordinates of the top, left corner of the rectangle.

3.5.2 Methods

void clearRect (int x, int y, int width, int height)

    Purpose: Clear the rectangle by filling it with the background surface.
void drawArc (int x, int y, int width, int height, int startAngle, int arcAngle)

Purpose: Draws the outline of an elliptical arc inscribed in the specified rectangle, starting at startAngle degrees (0 is 3 o’clock), for arcAngle degrees (positive is anticlockwise).

void drawLine (int x1, int y1, int x2, int y2)

Purpose: Draw a line from (x1, y1) to (x2, y2).

void drawOval (int x, int y, int width, int height)

Purpose: Draws the outline of an oval inscribed in the specified rectangle.

void drawPolygon (int[] xpoints, int[] ypoints, int npoints)

Purpose: Draws the outline of a closed polygon with npoints vertices with coordinates (xpoints[i], ypoints[i]).

void drawPolyline (int[] xpoints, int[] ypoints, int npoints)

Purpose: Draws a sequence of npoints − 1 connected lines with npoints end points at coordinates (xpoints[i], ypoints[i]).

void drawRect (int x, int y, int width, int height)

Purpose: Draws the outline of the specified rectangle.

void drawRoundRect (int x, int y, int width, int height, int arcWidth, int arcHeight)

Purpose: Draws the outline of the specified rectangle with rounded corners made with arcs with width arcWidth and height arcHeight.

void drawString (String s, int x, int y)

Purpose: Draw string s with (x, y) being the coordinates of the bottom-left corner of the first character.

void fillArc (int x, int y, int width, int height, int startAngle, int arcAngle)

Purpose: Fills an elliptical arc inscribed in the specified rectangle, starting at startAngle degrees (0 is 3 o’clock), for arcAngle degrees (positive is anticlockwise).

void fillOval (int x, int y, int width, int height)
void fillPolygon (int[] xpoints, int[] ypoints, int npoints)

Purpose: Fills a closed polygon with npoints vertices with coordinates (xpoints[i], ypoints[i]).

void fillRect (int x, int y, int width, int height)

Purpose: Fills the specified rectangle.

void fillRoundRect (int x, int y, int width, int height, int arcWidth, int arcHeight)

Purpose: Fills the specified rectangle with rounded corners made with arcs with width arcWidth and height arcHeight.

void setColor (int red, int green, int blue)

Purpose: Set the color to draw with by defining the amounts of (red, green, blue). (0, 0, 0) is black. (255, 255, 255) is white.

Precondition: 0 ≤ red ≤ 255; 0 ≤ green ≤ 255; and 0 ≤ blue ≤ 255.

void drawImage (int imageID, int x, int y)

Purpose: Draw the image selected by imageID with (x, y) being the coordinates of its top-left corner.

Precondition: imageID must refer to an image that has been preloaded with getImage(String).

3.6 Loading images

3.6.1 Methods

int getImage (String path)

Purpose: Load an image from the file at path and return its new ID. If the ID returned is negative, the load failed.

3.7 Animation

3.7.1 Methods

void repaint ()

Purpose: Causes the window to be repainted.

void animate (int fps)

Purpose: Causes the window to be repainted at fps frames per second. Only call this method once.

Precondition: fps > 0.
3.8 Sounds

3.8.1 Purpose

Like images, sound files can be loaded and played many times. File formats include .wav and .aif. Others may work. Loaded sounds are identified by an ID number.

A sound once loaded, can be played or looped. If it is played again, before it is finished, it will start from the beginning again. Multiple sounds may be played at the same time. A single sound file loaded multiple times may be played at the same time.

3.8.2 Methods

int getSound (String url)

*Purpose:* Load a sound clip from the file at *url* and return its new ID. If the ID returned is negative, the load failed.

*Precondition:* *url* is a URL, not just a file name. For example, file:sound.wav.

void playSound (int soundID)

*Purpose:* Play the sound selected by *soundID*.

*Precondition:* *soundID* must refer to a sound that has been preloaded with getSound(String).

void loopSound (int soundID)

*Purpose:* Loop the sound selected by *soundID*.

*Precondition:* *soundID* must refer to a sound that has been preloaded with getSound(String).

void stopSound (int soundID)

*Purpose:* Stop the sound selected by *soundID*.

*Precondition:* *soundID* must refer to a sound that has been preloaded with getSound(String).

3.9 Using the mouse

3.9.1 Purpose

Implement the rewrite methods to handle mouse events that occur within the content area of the frame. The functions provide the mouse position at any time.
3.9.2 Rewrites

void onMouseClicked (int x, int y)

*Purpose:* Implement this method to handle mouse clicks in the content area of the frame.

*(x, y)* will be the passed the coordinates of the mouse relative to the content area of the frame at the time the event occurred.

void onMouseEntered (int x, int y)

*Purpose:* Implement this method to handle the mouse entering the content area of the frame.

*(x, y)* will be the passed the coordinates of the mouse relative to the content area of the frame at the time the event occurred.

void onMouseExited (int x, int y)

*Purpose:* Implement this method to handle the mouse exiting the content area of the frame.

*(x, y)* will be the passed the coordinates of the mouse relative to the content area of the frame at the time the event occurred.

void onMousePressed (int x, int y)

*Purpose:* Implement this method to handle the mouse being pressed within the content area of the frame.

*(x, y)* will be the passed the coordinates of the mouse relative to the content area of the frame at the time the event occurred.

void onMouseReleased (int x, int y)

*Purpose:* Implement this method to handle the mouse being released within the content area of the frame.

*(x, y)* will be the passed the coordinates of the mouse relative to the content area of the frame at the time the event occurred.

3.9.3 Methods

int mouseX ()

*Purpose:* Returns the current horizontal ordinate *x* of the mouse pointer relative to the content area of the frame.

int mouseY ()

*Purpose:* Returns the current vertical ordinate *y* of the mouse pointer relative to the content area of the frame.
3.10 Keyboard events

3.10.1 Purpose

Implement the rewrite methods to handle keyboard events when the frame is in focus.

The codes that identify keys may be discovered by experiment. (Write handlers that print the codes to the console.) Some handy codes are those for the arrow keys: left = 37; up = 38; right = 39; and down = 40.

3.10.2 Rewrites

void onKeyPressed (int code)

Purpose: Implement this method to handle a key being pressed while the frame is in focus.

code will be the passed a code that identifies the key.

void onKeyReleased (int code)

Purpose: Implement this method to handle a key being released while the frame is in focus.

code will be the passed a code that identifies the key.

void onKeyTyped (char c)

Purpose: Implement this method to handle a key being typed while the frame is in focus.

c will be the passed the character that was typed.

3.11 Waiting for fixed times

3.11.1 Methods

void sleep (int ms)

Purpose: Makes the program wait for a requested number of ms (miliseconds).

3.12 Console input

3.12.1 Purpose

The following are methods for reading from standard input (which is usually the keyboard) and for checking to see if there is more input to read.
3.12.2 Methods

int readInt ()

_Purpose_: Returns the next integer from standard input.

_Precondition_: Will cause a run time error unless there is an integer in standard input to read. Use _isNextInt_() to check first.

long readLong ()

_Purpose_: Returns the next long from standard input.

_Precondition_: Will cause a run time error unless there is a long in standard input to read. Use _isNextLong_() to check first.

boolean readBoolean ()

_Purpose_: Returns the next boolean from standard input.

_Precondition_: Will cause a run time error unless there is a boolean in standard input to read. Use _isNextBoolean_() to check first.

double readDouble ()

_Purpose_: Returns the next double from standard input.

_Precondition_: Will cause a run time error unless there is a double in standard input to read. Use _isNextDouble_() to check first.

float readFloat ()

_Purpose_: Returns the next float from standard input.

_Precondition_: Will cause a run time error unless there is a float in standard input to read. Use _isNextFloat_() to check first.

String readWord ()

_Purpose_: Returns the next word as a _String_. A “word” is a sequence of one-or-more non-whitespace characters.

_Precondition_: Will cause a run time error unless there is a word in standard input to read. Use _isNextWord_() to check first.

String readLine ()

_Purpose_: Returns the next line of text as a _String_. A line is a sequence of zero-or-more characters terminated by the end of line, which is not returned as part of the line.
Precondition: Will cause a run time error unless there is a line in standard input to read. Use isNextLine() to check first.

`boolean isNextInt()`

*Purpose:* Returns `true` if and only if there is an integer in standard input available to read, that is `readInt()` would succeed.

`boolean isNextLong()`

*Purpose:* Returns `true` if and only if there is a long in standard input available to read, that is `readLong()` would succeed.

`boolean isNextBoolean()`

*Purpose:* Returns `true` if and only if there is an boolean in standard input available to read, that is `readBoolean()` would succeed.

`boolean isNextDouble()`

*Purpose:* Returns `true` if and only if there is a double in standard input available to read, that is `readDouble()` would succeed.

`boolean isNextFloat()`

*Purpose:* Returns `true` if and only if there is a float in standard input available to read, that is `readFloat()` would succeed.

`boolean isNextWord()`

*Purpose:* Returns `true` if and only if there is a word in standard input available to read, that is `readWord()` would succeed.

`boolean isNextLine()`

*Purpose:* Returns `true` if and only if there is a line in standard input available to read, that is `readLine()` would succeed.

### 3.13 Console output

#### 3.13.1 Purpose

The following are methods for printing to standard output (which is usually the terminal window).
3.13.2 Methods

void print (char c)

    Purpose: Writes character c to standard output.

void print (String s)

    Purpose: Writes string s to standard output.

void print (long i)

    Purpose: Writes integral i to standard output.

void print (boolean p)

    Purpose: Writes boolean p to standard output.

void print (double x)

    Purpose: Writes floating point number x to standard output.

void println (char c)

    Purpose: Writes character c and then a newline to standard output.

void println ()

    Purpose: Writes a newline to standard output.

void println (String s)

    Purpose: Writes string s and then a newline to standard output.

void println (long i)

    Purpose: Writes integral i and then a newline to standard output.

void println (boolean p)

    Purpose: Writes boolean p and then a newline to standard output.

void println (double x)

    Purpose: Writes floating point number x and then a newline to standard output.

3.14 Math

3.14.1 Purpose

The following are some commonly used numeric constants and functions.
3.14.2 Constants

final int MAX_INT

Purpose: A constant holding the maximum value an int can have, $2^{31} - 1$.

final int MIN_INT

Purpose: A constant holding the minimum value an int can have, $-2^{31}$.

final long MAX_LONG

Purpose: A constant holding the maximum value a long can have, $2^{63} - 1$.

final long MIN_LONG

Purpose: A constant holding the minimum value a long can have, $-2^{63}$.

final double PI

Purpose: The closest double approximation to π.

final double E

Purpose: The closest double approximation to e.

3.14.3 Methods

double abs (double a)

Purpose: Returns the absolute value of a.

float abs (float a)

Purpose: Returns the absolute value of a.

long abs (long a)

Purpose: Returns the absolute value of a.

int abs (int a)

Purpose: Returns the absolute value of a.

double ceil (double a)
Purpose: Returns the least double value that is greater than or equal to \( a \) and equal to an integer.

double exp (double x)

Purpose: Returns \( e^x \), that is Euler’s constant \( e \) raised to power \( x \).

double floor (double a)

Purpose: Returns the greatest double value that is less than or equal to \( a \) and equal to an integer.

double log (double x)

Purpose: Returns the natural logarithm of \( x \).

double rint (double x)

Purpose: Returns the closest mathematical integer to \( x \).

long round (double a)

Purpose: Returns the closest long to \( a \).

int round (float a)

Purpose: Returns the closest int to \( a \).

double sqrt (double a)

Purpose: Returns the square root of \( a \).

Precondition: \( a \geq 0.0 \).

double pow (double a, double b)

Purpose: Returns \( a \) raised to the power \( b \), \( a^b \).

double sin (double a)

Purpose: Returns the trigonometric sine of \( a \) radians.

double cos (double a)

Purpose: Returns the trigonometric cosine of \( a \) radians.

double tan (double a)

Purpose: Returns the trigonometric tangent of \( a \) radians.
double asin (double a)

*Purpose:* Returns the trigonometric arc sine of a in radians.

double acos (double a)

*Purpose:* Returns the trigonometric arc cosine of a in radians.

double atan (double a)

*Purpose:* Returns the trigonometric arc tangent of a in radians.

double atan2 (double y, double x)

*Purpose:* Returns the angle theta from the conversion of rectangular coordinates (x, y) to polar coordinates (r, theta).

double max (double a, double b)

*Purpose:* Returns the greater of a and b.

float max (float a, float b)

*Purpose:* Returns the greater of a and b.

int max (int a, int b)

*Purpose:* Returns the greater of a and b.

long max (long a, long b)

*Purpose:* Returns the greater of a and b.

double min (double a, double b)

*Purpose:* Returns the lesser of a and b.

float min (float a, float b)

*Purpose:* Returns the lesser of a and b.

int min (int a, int b)

*Purpose:* Returns the lesser of a and b.

long min (long a, long b)

*Purpose:* Returns the lesser of a and b.

double random ()

*Purpose:* Returns a random value x such that 0.0 ≤ x < 1.0.
3.15 Strings

3.15.1 Purpose

The following are methods for working with Strings.

3.15.2 Methods

int length (String s)

    Purpose: Returns the length of s.

cchar charAt (String s, int i)

    Purpose: Returns the character at position i in s.

    Precondition: 0 ≤ i < length(s).

boolean equals (String a, String b)

    Purpose: Returns true if and only if a contains the same sequence of characters as in b.

String format (boolean p, int w)

    Purpose: Returns p converted to a string, padded with spaces to a minimum width |w|. If w is negative, the result is left-justified, otherwise right-justified.

    Examples:
    format(true, 10) returns " true"; and
    format(true, -10) returns "true ".

String format (char c, int w)

    Purpose: Returns c converted to a string, padded with spaces to a minimum width |w|. If w is negative, the result is left-justified, otherwise right-justified.

    Examples:
    format(’a’, 3) returns " a"; and
    format(’a’, -3) returns "a ".

String format (double d, int w, char f, int p)

    Purpose: Returns d converted to a string, padded with spaces to a minimum width |w|. If w is negative, the result is left-justified, otherwise right-justified. f controls the format: ’e’ selects scientific notation; ’f’ selects fixed point; or ’g’ selects the best format depending on the number and p. For ’e’ and ’f’, p is the number
of decimal digits to display after the decimal point, but for 'g' it is the total number of digits.

Examples:
format(1234.56789, 12, 'e', 4) returns " 1.2346e+03";
format(1234.56789, 12, 'f', 4) returns " 1234.5679";
format(1234.56789, 12, 'g', 4) returns " 1235";
format(0.000001234567, 12, 'e', 4) returns " 1.2346e-06";
format(0.000001234567, 12, 'f', 4) returns " 0.0000";
and
format(0.000001234567, 12, 'g', 4) returns " 1.235e-06".

String format (long l, int w)

*Purpose:* Returns l converted to a string, padded with spaces to a minimum width |w|. If w is negative, the result is left-justified, otherwise right-justified.

Examples:
format(42, 5) returns " 42"; and
format(42, -5) returns "42 ".

String format (String s, int w)

*Purpose:* Returns s padded with spaces to a minimum width |w|. If w is negative, the result is left-justified, otherwise right-justified.

Examples:
format("aaa", 5) returns " aaa"; and
format("aaa", -5) returns "aaa ".

boolean parseBoolean (String s)

*Purpose:* Returns s converted to a boolean.

int parseInt (String s)

*Purpose:* Returns s converted to an int.

long parseLong (String s)

*Purpose:* Returns s converted to a long.

float parseFloat (String s)

*Purpose:* Returns s converted to a float.

double parseDouble (String s)

*Purpose:* Returns s converted to a double.
3.16 Terminating a program

3.16.1 Methods

void exit ()

*Purpose:* Terminates the program.

3.17 Debugger

3.17.1 Purpose

This environment supports a debugger with a graphical user interface. It allows the user to slow down the execution of a program, so that the flow of control through the program may be observed and the values in variables monitored.

To activate the debugger, compile your program with the `mashc +debug` option.

While the program is running under the control of the debugger, the original MaSH program’s source code is displayed on the left, with a pointer showing the statement that is about to execute. Sometimes it will point to a closing brace, indicating that the current method is about to be exited, or a loop guard is about to be tested again, depending on the context.

The contents of all the program’s variables are displayed on the right in a table. The entries in the `scope` column, consist of a number and a word. The number indicates which method invocation the variable belongs to. For no-method MaSH programs the number is always 0. For method MaSH programs, 0 indicates a global variable, 1 indicates the `main` method, 2 indicates a method called from `main`, 3 indicates a method called from 2, etc. The word is blank for variables in a no-method program, unless the variable is declared in a block, which makes it `local`. In method programs, the word can be: `global`; `local`; or `param`. Variables are added to the table as they come into scope and removed again when their enclosing block exits. Anything changed on the last program step is displayed in green.

Limitations: The debugger can not display values of arrays with 3 or more dimensions. The debugger can not step through methods that run in the program’s event dispatch thread (for example, the `paintWindow` method in the `graphics` environment). This GUI debugger can not work in environments without a big screen (for example, Lego Mindstorms).

3.17.2 Methods

void suspendDebugger ()

*Purpose:* Suspend the debugger.

Calling this in your MaSH program disables the debugger from pausing your program or updating the debugger window.
If the debugger has not been included at compile time, this procedure does nothing at all.

```java
void resumeDebugger ()
```

*Purpose:* Resume the debugger.

Calling this in your MaSH program re-enables the debugger to pause your program and update the debugger window.

If the debugger has not been included at compile time, this procedure does nothing at all.

Do not call this in methods that run in the program’s event dispatch thread (for example, the `paintWindow` method in the `graphics` environment).
4 Environment nxt

4.1 Purpose

This environment supports programming a Lego Mindstorms NXT robot, via the Lejos NXJ system.

Requires Lejos-NXJ version 0.9.1.

4.2 Rewrites

mandatory

void main ()

Purpose: A program that is organised into methods must have a main method (a procedure with no arguments). This will be the first method to execute. mashc automatically rewrites this method to conform to standard Java.

4.3 Setting up sensors

4.3.1 Purpose

At the start of a program, the sensor ports need to be set up to work with the kind of sensor that is plugged into it. Sensors can be either the old RCX kinds or the new NXT kinds.

The rotation sensor built into an NXT motor does not have to be set up.

Light, sound and proximity sensors benefit from a time delay of about half a second between setting the sensor up and making measurements.

4.3.2 Constants

final int TOUCH

Purpose: Constant to select sensor type touch (NXT or RCX).

final int LIGHT_FLOOD

Purpose: Constant to select sensor type light (NXT with the flood-light on).

final int LIGHT_NOFLOOD

Purpose: Constant to select sensor type light (NXT with the flood-light off).

final int LIGHT_RCX

Purpose: Constant to select sensor type light (RCX with the flood-light on).
final int COLOR

Purpose: Constant to select sensor type color (NXT), initally with its floodlight off.

final int ROTATION_RCX

Purpose: Constant to select sensor type rotation (RCX).

final int SOUND

Purpose: Constant to select sensor type sound (NXT).

final int PROXIMITY

Purpose: Constant to select sensor type ultrasonic proximity (NXT).

4.3.3 Methods

void setUpSensor (int port, int type)

Purpose: Sets up the port to be a sensor of the given type.

Precondition: port is 1, 2, 3, or 4.

Precondition: type is TOUCH, LIGHT_FLOOD, LIGHT_NOFLOOD, LIGHT_RCX, COLOR, ROTATION_RCX, SOUND, or PROXIMITY.

4.4 Output port constants

4.4.1 Constants

final int A

Purpose: Constant to select port A.

final int B

Purpose: Constant to select port B.

final int C

Purpose: Constant to select port C.

4.5 Using touch sensors

4.5.1 Purpose

These methods provide either waits for a touch sensor’s state to change or the current state of the touch sensor.

These methods all assume that the port number you provide as a parameter has been set up as a touch sensor of either the RCX or NXT kind.
4.5.2 Methods

void waitForPush (int port)

*Purpose:* Makes the program wait until the touch sensor on *port* is pushed.

*Precondition:* *port* is 1, 2, 3 or 4.

void waitForLetGo (int port)

*Purpose:* Makes the program wait until the touch sensor on *port* is let go.

*Precondition:* *port* is 1, 2, 3, or 4.

boolean isPushed (int port)

*Purpose:* Returns true if and only if the touch sensor on *port* is currently pushed.

*Precondition:* *port* is 1, 2, 3, or 4.

4.6 Using light sensors

4.6.1 Purpose

Most of these methods provide waits for the light level reported by a light sensor to change. In addition the current light level reported by a light sensor can be obtained and (for NXT sensors) the floodlight turned on or off.

These methods all assume that the *port* number you provide as a parameter has been set up as a light sensor, of either the RCX or NXT kind, or as a colour sensor.

4.6.2 Methods

void waitForLighter (int port, int dif)

*Purpose:* Makes the program wait until the light sensor reading on *port* is increased by *dif*.

*Precondition:* *port* is 1, 2, 3 or 4.

*Precondition:* *dif* is between 0 and 100, inclusive. 0 is no wait at all. Real light levels never really change by anything like 100.

void waitForLight (int port, int light)

*Purpose:* Makes the program wait until the light sensor reading on *port* is at least the desired *light* level.
Precondition: port is 1, 2, 3 or 4.

Precondition: light is between 0 and 100, inclusive.

void waitForDarker (int port, int dif)

Purpose: Makes the program wait until the light sensor reading on port is decreased by dif.

Precondition: port is 1, 2, 3 or 4.

Precondition: dif is between 0 and 100, inclusive. 0 is no wait at all. Real light levels never really change by anything like 100.

void waitForDark (int port, int light)

Purpose: Makes the program wait until the light sensor reading on port is at most the desired light level.

Precondition: port is 1, 2, 3 or 4.

Precondition: light is between 0 and 100, inclusive.

int getLight (int port)

Purpose: Returns the current light sensor reading on port.
The reading will depend on whether the floodlight is on or not, and what colour it is.

Precondition: port is 1, 2, 3 or 4.

void setFloodlight (int port, boolean floodlight)

Purpose: Turn the floodlight of the light sensor on port on or off.
For the colour sensor, turning it on, sets it to red.

Precondition: port is 1, 2, 3 or 4.

4.7 Using colour sensors

4.7.1 Purpose

These constants and methods are those specific to the NXT colour sensor. All of the light sensor methods can also be used with the colour sensor.

These methods all assume that the port number you provide as a parameter has been set up as a colour sensor.
4.7.2 Constants

final int BLACK

Purpose: ID code for a detectable colour.

final int BLUE

Purpose: ID code for a detectable colour and for selecting the floodlight colour.

final int GREEN

Purpose: ID code for a detectable colour and for selecting the floodlight colour.

final int NONE

Purpose: Constant to select the floodlight colour (off).

final int RED

Purpose: ID code for a detectable colour and for selecting the floodlight colour.

final int WHITE

Purpose: ID code for a detectable colour.

4.7.3 Methods

void setFloodlight (int port, int color)

Purpose: Turn the floodlight of the colour sensor at port on with the selected color, or off with NONE.

Precondition: port is 1, 2, 3 or 4.

Precondition: color is BLUE, GREEN, RED, or NONE (off).

int getColor (int port)

Purpose: This function automatically controls the floodlight colour and reads light levels and decides which colour is under the colour sensor at port. It returns one of BLACK, BLUE, GREEN, RED, or WHITE.

Precondition: port is 1, 2, 3 or 4.
4.8 Using sound sensors

4.8.1 Purpose

These methods provide either waits for the sound level reported by a sound sensor to change or the current volume reported by the sound sensor.

These methods all assume that the port number you provide as a parameter has been set up as a sound sensor.

4.8.2 Methods

void waitForLouder (int port, int dif)

Purpose: Makes the program wait until the sound sensor reading on port is increased by dif.

Precondition: port is 1, 2, 3 or 4.

Precondition: dif is between 0 and 100, inclusive. 0 is no wait at all.

void waitForLoud (int port, int volume)

Purpose: Makes the program wait until the sound sensor reading on port is at least the desired volume level.

Precondition: port is 1, 2, 3 or 4.

Precondition: volume is between 0 and 100, inclusive.

void waitForQuieter (int port, int dif)

Purpose: Makes the program wait until the sound sensor reading on port has decreased by dif.

Precondition: port is 1, 2, 3 or 4.

Precondition: dif is between 0 and 100, inclusive. 0 is no wait at all.

void waitForQuiet (int port, int volume)

Purpose: Makes the program wait until the sound sensor reading on port is at most the desired volume level.

Precondition: port is 1, 2, 3 or 4.

Precondition: volume is between 0 and 100, inclusive.

int getVolume (int port)

Purpose: Returns the current volume sensor reading on port.

Precondition: port is 1, 2, 3 or 4.
4.9 Using ultrasonic sensors

4.9.1 Purpose

These methods provide either waits for the distance reported by an ultrasonic sensor to change or the current distance reported by the ultrasonic sensor.

These methods all assume that the port number you provide as a parameter has been set up as an ultrasonic sensor.

A known bug in Lejos 0.85 means the ultrasonic sensor does not work in input port 4.

4.9.2 Methods

void waitForNearer (int port, int dif)

Purpose: Makes the program wait until the Ultrasonic sensor on port senses that distance has decreased by dif cm.

Precondition: port is 1, 2, 3 or 4.

Precondition: dif is between 0 and 255, inclusive. 0 is no wait at all.

void waitForNear (int port, int distance)

Purpose: Makes the program wait until the ultrasonic sensor on port reports a distance which is at most distance cm. A distance of 255 means no object is in view.

Precondition: port is 1, 2, 3 or 4.

Precondition: distance is between 0 and 255, inclusive.

void waitForFurther (int port, int dif)

Purpose: Makes the program wait until the ultrasonic sensor on port reports the distance has increased by dif cm.

Precondition: port is 1, 2, 3 or 4.

Precondition: dif is between 0 and 255, inclusive. 0 is no wait at all. 255 represents no object currently in view.

void waitForFar (int port, int distance)

Purpose: Makes the program wait until the ultrasonic sensor on port reports a distance which is at least distance cm. A distance of 255 means no object is in view.
Precondition: port is 1, 2, 3 or 4.

Precondition: distance is between 0 and 255, inclusive.

int getDistance (int port)

Purpose: Returns the current ultrasonic sensor reading on port in centimetres. 255 means no object is in view.

Precondition: port is 1, 2, 3 or 4.

4.10 Using rotation sensors

4.10.1 Purpose

These methods provide a wait for the rotation count reported by a rotation sensor to change, the current rotation count reported by a sensor, or reset the count to zero.

An NXT rotation sensor registers 360 counts per full 360 degree rotation. NXT rotation sensors are built into NXT motors and must be plugged into port A, B or C. There is no need to call setUpSensor(int,int) to set up these sensors.

An RCX rotation sensor registers 16 counts per full 360 degree rotation. It must be plugged into port 1, 2, 3 or 4 and set up like other sensor kinds.

4.10.2 Methods

void resetRotation (int port)

Purpose: Sets the counter in the rotation sensor on port to zero.

Precondition: port is 1, 2, 3, 4, A, B, or C.

void waitForRotation (int port, int rotation)

Purpose: Makes the program wait until the counter in the rotation sensor on port has changed by at least the absolute value of rotation.

This method does not reset the counter in the rotation sensor.

Precondition: port is 1, 2, 3, 4, A, B, or C.

int getRotation (int port)

Purpose: Returns the current rotation sensor reading from port.

Precondition: port is 1, 2, 3, 4, A, B, or C.
4.11 Using motors

4.11.1 Purpose

The following methods apply to both NXT and RCX motors.

4.11.2 Methods

void motorForward (int port, int power)

*Purpose*: Make the motor on port go forwards at the given power.

*Precondition*: port is A, B, or C.

*Precondition*: power is between 0 and 100, inclusive.

void motorBackward (int port, int power)

*Purpose*: Make the motor on port go backwards at the given power.

*Precondition*: port is A, B, or C.

*Precondition*: power is between 0 and 100, inclusive.

void motorStop (int port)

*Purpose*: Stop the motor on port. Stopping a motor cuts the power to it and stops it rotating.

*Precondition*: port is A, B, or C.

void motorFloat (int port)

*Purpose*: Float the motor on port. Floating a motor, cuts power to it and lets it rotate freely.

*Precondition*: port is A, B, or C.

4.12 Using lamps

4.12.1 Methods

void lampOn (int port, int power)

*Purpose*: Make the lamp on port go on at the given power.

*Precondition*: port is A, B, or C.

*Precondition*: power is between 0 and 100, inclusive.

void lampOff (int port)

*Purpose*: Turns the lamp on port off.

*Precondition*: port is A, B, or C.
4.13 Making sounds

4.13.1 Constants

final int[] PIANO

Purpose: Attack, decay, sustain and release shape parameters to emulate a piano. Use with playNote(int[], int, int).

final int[] FLUTE

Purpose: Attack, decay, sustain and release shape parameters to emulate a flute. Use with playNote(int[], int, int).

final int[] XYLOPHONE

Purpose: Attack, decay, sustain and release shape parameters to emulate a xylophone. Use with playNote(int[], int, int).

4.13.2 Methods

void beep ()

Purpose: Beep once.

void twoBeeps ()

Purpose: Beep twice.

void fallingBeeps ()

Purpose: A few short beeps with descending tones.

void risingBeeps ()

Purpose: A few short beeps with ascending tones.

void buzz ()

Purpose: Makes a low buzz.

void pause (int t)

Purpose: Make no sound for t milliseconds.

void playTone (int frequency, int duration)

Purpose: Plays a tone, given its frequency (Hertz) and duration (milliseconds).

void playNote (int[] instrument, int frequency, int duration)

Purpose: Plays a note attack, decay, sustain and release shape specified by a selected instrument, given its frequency (Hertz) and duration (milliseconds).
4.14 Using the LCD

4.14.1 Purpose
The LCD display can be used to display text and graphics.

4.14.2 Constants

final int LCD_WIDTH

Purpose: The width of the LCD display in pixels, equals 100.

final int LCD_HEIGHT

Purpose: The height of the LCD display in pixels, equals 64.

final int LCD_COLUMNS

Purpose: The width of the LCD display in character spaces, equals 16.

final int LCD_LINES

Purpose: The height of the LCD display in character spaces, equals 8.

4.14.3 Methods

void clear ()

Purpose: Clears the LCD display to all white.

void setPixel (boolean p, int x, int y)

Purpose: Set the pixel at (x, y). If p is true, the pixel is set to black, white otherwise.

Precondition: 0 ≤ x < LCD_WIDTH.

Precondition: 0 ≤ y < LCD_HEIGHT.

void drawString (String s, int x, int y, boolean invert)

Purpose: Displays a string s on the LCD starting at (x, y). If invert is true then it is drawn as white on black, instead of the usual black on white.

Precondition: 0 ≤ x < LCD_COLUMNS.

Precondition: 0 ≤ y < LCD_LINES.
void drawString (String s, int x, int y)

    Purpose: Displays a string s on the LCD starting at column x and line y.

    Precondition: 0 ≤ x < LCD_COLUMNS.

    Precondition: 0 ≤ y < LCD_LINES.

void drawInt (int i, int width, int x, int y)

    Purpose: Displays integer i on the LCD starting at column x and line y right justified within at least width spaces.

    Precondition: 1 ≤ width ≤ 11.

    Precondition: 0 ≤ x < LCD_COLUMNS.

    Precondition: 0 ≤ y < LCD_LINES.

void scroll ()

    Purpose: Scrolls the screen up one text line.

4.15  Using buttons

4.15.1  Purpose

There are four buttons on the front of an NXT. These rewrites define handers for pressing on or releasing the ENTER, LEFT, RIGHT and ESCAPE buttons.

    Usually a button should cause an action when it is released, so actions are usually put in an onRelease handler, but programs can also do something while the button is pressed by implementing an onPress handler.

    Note: Releasing the ESCAPE button, by default, is the easiest way for users to terminate a program with an endless loop. Think carefully before changing that behaviour.

4.15.2  Rewrites

void onPressEnter ()

    Purpose: Write a procedure of this type to handle presses on the ENTER button (the orange one).

void onReleaseEnter ()

    Purpose: Write a procedure of this type to handle releases of the ENTER button.
void onPressLeft ()

*Purpose:* Write a procedure of this type to handle presses on the LEFT button.

void onReleaseLeft ()

*Purpose:* Write a procedure of this type to handle releases of the LEFT button.

void onPressRight ()

*Purpose:* Write a procedure of this type to handle presses on the RIGHT button.

void onReleaseRight ()

*Purpose:* Write a procedure of this type to handle releases of the RIGHT button.

void onPressEscape ()

*Purpose:* Write a procedure of this type to handle presses on the ESCAPE button.

void onReleaseEscape ()

*Purpose:* Write a procedure of this type to handle releases of the ESCAPE button.

4.16 Waiting for fixed times

4.16.1 Methods

void sleep (int ms)

*Purpose:* Makes the program wait for a requested number of ms (miliseconds).

4.17 Terminating a program

4.17.1 Methods

void exit ()

*Purpose:* Terminates the program.
4.18 Math

4.18.1 Purpose
The following are some commonly used numeric constants and functions.

4.18.2 Constants

final int MAX_INT

*Purpose*: A constant holding the maximum value an `int` can have, $2^{31} - 1$.

final int MIN_INT

*Purpose*: A constant holding the minimum value an `int` can have, $-2^{31}$.

final long MAX_LONG

*Purpose*: A constant holding the maximum value a `long` can have, $2^{63} - 1$.

final long MIN_LONG

*Purpose*: A constant holding the minimum value a `long` can have, $-2^{63}$.

final double PI

*Purpose*: The closest `double` approximation to $\pi$.

final double E

*Purpose*: The closest `double` approximation to $e$.

4.18.3 Methods

double abs (double a)

*Purpose*: Returns the absolute value of a.

float abs (float a)

*Purpose*: Returns the absolute value of a.

long abs (long a)

*Purpose*: Returns the absolute value of a.
int abs (int a)

Purpose: Returns the absolute value of a.

double ceil (double a)

Purpose: Returns the least double value that is greater than or equal to a and equal to an integer.

double exp (double x)

Purpose: Returns $e^x$, that is Euler’s constant $e$ raised to power $x$.

double floor (double a)

Purpose: Returns the greatest double value that is less than or equal to a and equal to an integer.

double log (double x)

Purpose: Returns the natural logarithm of $x$.

double rint (double x)

Purpose: Returns the closest mathematical integer to $x$.

long round (double a)

Purpose: Returns the closest long to a.

int round (float a)

Purpose: Returns the closest int to a.

double sqrt (double a)

Purpose: Returns the square root of a.

Precondition: $a \geq 0.0$.

double pow (double a, double b)

Purpose: Returns $a$ raised to the power $b$, $a^b$.

double sin (double a)

Purpose: Returns the trigonometric sine of $a$ radians.

double cos (double a)
**double tan (double a)**

*Purpose:* Returns the trigonometric tangent of a radians.

**double asin (double a)**

*Purpose:* Returns the trigonometric arc sine of a in radians.

**double acos (double a)**

*Purpose:* Returns the trigonometric arc cosine of a in radians.

**double atan (double a)**

*Purpose:* Returns the trigonometric arc tangent of a in radians.

**double atan2 (double y, double x)**

*Purpose:* Returns the angle $\theta$ from the conversion of rectangular coordinates $(x, y)$ to polar coordinates $(r, \theta)$.

**double max (double a, double b)**

*Purpose:* Returns the greater of a and b.

**float max (float a, float b)**

*Purpose:* Returns the greater of a and b.

**int max (int a, int b)**

*Purpose:* Returns the greater of a and b.

**long max (long a, long b)**

*Purpose:* Returns the greater of a and b.

**double min (double a, double b)**

*Purpose:* Returns the lesser of a and b.

**float min (float a, float b)**

*Purpose:* Returns the lesser of a and b.

**int min (int a, int b)**

*Purpose:* Returns the lesser of a and b.

**long min (long a, long b)**

*Purpose:* Returns the lesser of a and b.

**double random ()**

*Purpose:* Returns a random value $x$ such that $0.0 \leq x < 1.0$. 
4.19 Strings

4.19.1 Purpose

The following are methods for working with Strings.

4.19.2 Methods

int length (String s)

*Purpose*: Returns the length of s.

char charAt (String s, int i)

*Purpose*: Returns the character at position i in s.

*Precondition*: $0 \leq i < \text{length}(s)$.

boolean equals (String a, String b)

*Purpose*: Returns true if and only if a contains the same sequence of characters as in b.

boolean parseBoolean (String s)

*Purpose*: Returns s converted to a boolean.

int parseInt (String s)

*Purpose*: Returns s converted to an int.

long parseLong (String s)

*Purpose*: Returns s converted to a long.

float parseFloat (String s)

*Purpose*: Returns s converted to a float.

double parseDouble (String s)

*Purpose*: Returns s converted to a double.

4.20 Threads

4.20.1 Purpose

A program may have up to 5 additional threads of execution.
4.20.2 Rewrites

void run1 ()

*Purpose:* Implement a run1() method that defines the actions of additional thread 1. Do no try to call it directly. To start this thread, call start1().

void run2 ()

*Purpose:* Implement a run2() method that defines the actions of additional thread 2. Do no try to call it directly. To start this thread, call start2().

void run3 ()

*Purpose:* Implement a run3() method that defines the actions of additional thread 3. Do no try to call it directly. To start this thread, call start3().

void run4 ()

*Purpose:* Implement a run4() method that defines the actions of additional thread 4. Do no try to call it directly. To start this thread, call start4().

void run5 ()

*Purpose:* Implement a run5() method that defines the actions of additional thread 5. Do no try to call it directly. To start this thread, call start5().

4.20.3 Methods

void start1 ()

*Purpose:* Call start1() to start additional thread 1.

*Precondition:* run1() has been implemented. If not, nxjc will report errors.

void start2 ()

*Purpose:* Call start2() to start additional thread 2.

*Precondition:* run2() has been implemented. If not, nxjc will report errors.

void start3 ()
Purpose: Call \texttt{start3()} to start additional thread 3.

Precondition: \texttt{run3()} has been implemented. If not, \texttt{nxjc} will report errors.

\begin{verbatim}
void start4 ()

    Purpose: Call \texttt{start4()} to start additional thread 4.

    Precondition: \texttt{run4()} has been implemented. If not, \texttt{nxjc} will report errors.
\end{verbatim}

\begin{verbatim}
void start5 ()

    Purpose: Call \texttt{start5()} to start additional thread 5.

    Precondition: \texttt{run5()} has been implemented. If not, \texttt{nxjc} will report errors.
\end{verbatim}

\section*{4.21 Sensor listeners}

\subsection*{4.21.1 Purpose}

An alternative to polling sensors is to register a listener that will respond when the sensor changes state.

\subsection*{4.21.2 Rewrites}

\begin{verbatim}
void onChange1 (int oldValue, int newValue)

    Purpose: Write a method with this type to handle changes on sensor port 1. \texttt{oldValue} and \texttt{newValue} are the old and new raw sensor values, respectively.
\end{verbatim}

\begin{verbatim}
void onChange2 (int oldValue, int newValue)

    Purpose: Write a method with this type to handle changes on sensor port 2. \texttt{oldValue} and \texttt{newValue} are the old and new raw sensor values, respectively.
\end{verbatim}

\begin{verbatim}
void onChange3 (int oldValue, int newValue)

    Purpose: Write a method with this type to handle changes on sensor port 3. \texttt{oldValue} and \texttt{newValue} are the old and new raw sensor values, respectively.
\end{verbatim}

\begin{verbatim}
void onChange4 (int oldValue, int newValue)

    Purpose: Write a method with this type to handle changes on sensor port 4. \texttt{oldValue} and \texttt{newValue} are the old and new raw sensor values, respectively.
\end{verbatim}
4.21.3 Methods

void listenPort1 ()

    *Purpose*: Starts listening to port 1.
    If the state of that sensor changes, \texttt{onChange1(int, int)} will be called.

void listenPort2 ()

    *Purpose*: Starts listening to port 2.
    If the state of that sensor changes, \texttt{onChange2(int, int)} will be called.

void listenPort3 ()

    *Purpose*: Starts listening to port 3.
    If the state of that sensor changes, \texttt{onChange3(int, int)} will be called.

void listenPort4 ()

    *Purpose*: Starts listening to port 4.
    If the state of that sensor changes, \texttt{onChange4(int, int)} will be called.
5 Environment rcx

5.1 Purpose

This environment supports programming a Lego Mindstorms RCX robot, via the Lejos system.

5.2 Rewrites

mandatory

void main ()

Purpose: A program that is organised into methods must have a main method (a procedure with no arguments). This will be the first method to execute. mashc automatically rewrites this method to conform to standard Java.

5.3 Setting up sensors

5.3.1 Constants

final int TOUCH

Purpose: Constant to select sensor type touch.

final int LIGHT

Purpose: Constant to select sensor type light.

final int ROTATION

Purpose: Constant to select sensor type rotation.

5.3.2 Methods

void setUpSensor (int port, int type)

Purpose: Sets up the port to be sensor of the given type.

Precondition: port is 1, 2, or 3.

Precondition: type is TOUCH, LIGHT, or ROTATION.
5.4 Using touch sensors

5.4.1 Methods

void waitForPush (int port)

*Purpose:* Makes the program wait until the touch sensor on `port` is pushed.

*Precondition:* `port` is 1, 2, or 3.

*Precondition:* The port has been set up as a touch sensor.

void waitForLetGo (int port)

*Purpose:* Makes the program wait until the touch sensor on `port` is let go.

*Precondition:* `port` is 1, 2, or 3.

*Precondition:* The port has been set up as a touch sensor.

boolean isPushed (int port)

*Purpose:* Returns `true` if and only if the button on `port` is currently pushed.

*Precondition:* `port` is 1, 2, or 3.

*Precondition:* The port has been set up as a touch sensor.

5.5 Using light sensors

5.5.1 Methods

void waitForLighter (int port, int dif)

*Purpose:* Makes the program wait until the light sensor reading on `port` is increased by `dif`.

*Precondition:* `port` is 1, 2, or 3.

*Precondition:* The port has been set up as a light sensor.

*Precondition:* `dif` is between 0 and 100, inclusive. 0 is no wait at all. Real light levels never really change by anything like 100.

void waitForLight (int port, int light)

*Purpose:* Makes the program wait until the light sensor reading on `port` is at least the desired `light` level.
Precondition: port is 1, 2, or 3.

Precondition: The port has been set up as a light sensor.

Precondition: light is between 0 and 100, inclusive.

```c
void waitForDarker (int port, int dif)
```

**Purpose:** Makes the program wait until the light sensor reading on port is decreased by dif.

*Precondition:* port is 1, 2, or 3.

*Precondition:* The port has been set up as a light sensor.

*Precondition:* dif is between 0 and 100, inclusive. 0 is no wait at all. Real light levels never really change by anything like 100.

```c
void waitForDark (int port, int light)
```

**Purpose:** Makes the program wait until the light sensor reading on port is at most the desired light level.

*Precondition:* port is 1, 2, or 3.

*Precondition:* The port has been set up as a light sensor.

*Precondition:* light is between 0 and 100, inclusive.

```c
int getLight (int port)
```

**Purpose:** Returns the current light sensor reading on port.

*Precondition:* port is 1, 2, or 3.

*Precondition:* The port has been set up as a light sensor.

### 5.6 Using rotation sensors

#### 5.6.1 Methods

```c
void waitForRotation (int port, int rotation)
```

**Purpose:** Makes the program wait until the counter in the rotation sensor on port has changed by at least the absolute value of rotation.

*Precondition:* port is 1, 2, or 3.

*Precondition:* The port has been set up as a rotation sensor.
int getRotation (int port)

    Purpose: Returns the current rotation sensor reading on port.

    Precondition: port is 1, 2, or 3.

    Precondition: The port has been set up as a rotation sensor.

void resetRotation (int port)

    Purpose: Sets the counter in the rotation sensor on port to zero.

    Precondition: port is 1, 2, or 3.

    Precondition: The port has been set up as a rotation sensor.

5.7 Output port constants

5.7.1 Constants

final int A

    Purpose: Constant to select port A.

final int B

    Purpose: Constant to select port B.

final int C

    Purpose: Constant to select port C.

5.8 Using motors

5.8.1 Methods

void motorForward (int port, int power)

    Purpose: Make the motor on port go forwards at the given power.

    Precondition: port is A, B, or C.

    Precondition: power is between 0 and 7, inclusive.

void motorBackward (int port, int power)

    Purpose: Make the motor on port go backwards at the given power.

    Precondition: port is A, B, or C.
Precondition: power is between 0 and 7, inclusive.

void motorStop (int port)

Purpose: Stop the motor on port.

Precondition: port is A, B, or C.

void motorFloat (int port)

Purpose: Float the motor on port.

Precondition: port is A, B, or C.

5.9 Using lamps
5.9.1 Methods

void lampOn (int port, int power)

Purpose: Make the lamp on port go on at the given power.

Precondition: port is A, B, or C.

Precondition: power is between 0 and 7, inclusive.

void lampOff (int port)

Purpose: Turns the lamp on port off.

Precondition: port is A, B, or C.

5.10 Waiting for fixed times
5.10.1 Methods

void sleep (int ms)

Purpose: Makes the program wait for a requested number of ms (miliseconds).

5.11 Making sounds
5.11.1 Methods

void systemSound (int i)
Purpose: Play system sound number i. The system sounds are as follows.

i  description
0  short beep
1  double beep
2  descending arpeggio
3  ascending arpeggio
4  long, low beep
5  quick ascending arpeggio

Precondition: 0 ≤ i ≤ 5.

void playTone (int frequency, int duration)

Purpose: Plays a tone, given its frequency (Hertz) and duration (centiseconds).

Precondition: 31 ≤ frequency ≤ 2100.

Precondition: 0 ≤ duration ≤ 256.

5.12 Using the LCD

5.12.1 Methods

void showNumber (int a)

Purpose: Displays a number in the LCD. Does not require refresh().

Precondition: 0 ≤ a ≤ 9999.

void clear ()

Purpose: Clears the LCD, but the effect will not show until refresh() is called.

void refresh ()

Purpose: Refreshes the LCD, causing the changes to be displayed.

void putChar (char c, int i)

Purpose: Puts a character c into the LCD at position i.

Precondition: 0 ≤ i ≤ 4.
5.13 Using infra-red communications

5.13.1 Methods

void sendByte (int i)

*Purpose:* Send one byte of information to another RCX, by infra-red transmission. The value to send, i, will be truncated if it can’t fit in a byte.

int receiveByte ()

*Purpose:* Waits for and returns a new byte sent by another RCX via infra-red transmission.

5.14 Math

5.14.1 Purpose

The following are some commonly used numeric constants and functions.

5.14.2 Constants

final int MAX_INT

*Purpose:* A constant holding the maximum value an int can have, \(2^{31} - 1\).

final int MIN_INT

*Purpose:* A constant holding the minimum value an int can have, \(-2^{31}\).

final double PI

*Purpose:* The closest double approximation to \(\pi\).

final double E

*Purpose:* The closest double approximation to \(e\).

5.14.3 Methods

double abs (double a)

*Purpose:* Returns the absolute value of a.

int abs (int a)

*Purpose:* Returns the absolute value of a.
double ceil (double a)

*Purpose:* Returns the least double value that is greater than or equal to a and equal to an integer.

double exp (double x)

*Purpose:* Returns $e^x$, that is Euler’s constant $e$ raised to power $x$.

double floor (double a)

*Purpose:* Returns the greatest double value that is less than or equal to a and equal to an integer.

double log (double x)

*Purpose:* Returns the natural logarithm of $x$.

double rint (double x)

*Purpose:* Returns the closest mathematical integer to $x$.

int round (float a)

*Purpose:* Returns the closest int to a.

double sqrt (double a)

*Purpose:* Returns the square root of $a$.  

*Precondition:* $a \geq 0.0$.

double pow (double a, double b)

*Purpose:* Returns $a$ raised to the power $b$, $a^b$.

double sin (double a)

*Purpose:* Returns the trigonometric sine of $a$ radians.

double cos (double a)

*Purpose:* Returns the trigonometric cosine of $a$ radians.

double tan (double a)

*Purpose:* Returns the trigonometric tangent of $a$ radians.

double asin (double a)
Purpose: Returns the trigonometric arc sine of \( a \) in radians.

double acos (double a)

Purpose: Returns the trigonometric arc cosine of \( a \) in radians.

double atan (double a)

Purpose: Returns the trigonometric arc tangent of \( a \) in radians.

double atan2 (double y, double x)

Purpose: Returns the angle \( \theta \) from the conversion of rectangular coordinates \( (x, y) \) to polar coordinates \( (r, \theta) \).

double max (double a, double b)

Purpose: Returns the greater of \( a \) and \( b \).

int max (int a, int b)

Purpose: Returns the greater of \( a \) and \( b \).

double min (double a, double b)

Purpose: Returns the lesser of \( a \) and \( b \).

int min (int a, int b)

Purpose: Returns the lesser of \( a \) and \( b \).

double random ()

Purpose: Returns a random value \( x \) such that \( 0.0 \leq x < 1.0 \).

5.15 Strings

5.15.1 Purpose

The following are methods for working with Strings.

5.15.2 Methods

int length (String s)

Purpose: Returns the length of \( s \).

char charAt (String s, int i)

Purpose: Returns the character at position \( i \) in \( s \).
Precondition: \( 0 \leq i < \text{length}(s) \).

boolean equals (String a, String b)

Purpose: Returns true if and only if a contains the same sequence of characters as in b.

boolean parseBoolean (String s)

Purpose: Returns s converted to a boolean.

int parseInt (String s)

Purpose: Returns s converted to an int.

long parseLong (String s)

Purpose: Returns s converted to a long.

float parseFloat (String s)

Purpose: Returns s converted to a float.

double parseDouble (String s)

Purpose: Returns s converted to a double.
6 Environment rcx2threads

6.1 Purpose

This environment supports programming a Lego Mindstorms RCX robot, via the Lejos system, for programs that require two execution threads.

6.2 Rewrites

mandatory
void main ()

Purpose: A program that is organised into methods must have a main method (a procedure with no arguments). This will be the first method to execute. mashc automatically rewrites this method to conform to standard Java.

6.3 Setting up sensors

6.3.1 Constants

final int TOUCH

Purpose: Constant to select sensor type touch.

final int LIGHT

Purpose: Constant to select sensor type light.

final int ROTATION

Purpose: Constant to select sensor type rotation.

6.3.2 Methods

void setUpSensor (int port, int type)

Purpose: Sets up the port to be sensor of the given type.

Precondition: port is 1, 2, or 3.

Precondition: type is TOUCH, LIGHT, or ROTATION.
6.4 Using touch sensors

6.4.1 Methods

void waitForPush (int port)

_Purpose:_ Makes the program wait until the touch sensor on _port_ is pushed.

_Precondition:_ _port_ is 1, 2, or 3.

_Precondition:_ The port has been set up as a touch sensor.

void waitForLetGo (int port)

_Purpose:_ Makes the program wait until the touch sensor on _port_ is let go.

_Precondition:_ _port_ is 1, 2, or 3.

_Precondition:_ The port has been set up as a touch sensor.

boolean isPushed (int port)

_Purpose:_ Returns _true_ if and only if the button on _port_ is currently pushed.

_Precondition:_ _port_ is 1, 2, or 3.

_Precondition:_ The port has been set up as a touch sensor.

6.5 Using light sensors

6.5.1 Methods

void waitForLighter (int port, int dif)

_Purpose:_ Makes the program wait until the light sensor reading on _port_ is increased by _dif_.

_Precondition:_ _port_ is 1, 2, or 3.

_Precondition:_ The port has been set up as a light sensor.

_Precondition:_ _dif_ is between 0 and 100, inclusive. 0 is no wait at all. Real light levels never really change by anything like 100.

void waitForLight (int port, int light)

_Purpose:_ Makes the program wait until the light sensor reading on _port_ is at least the desired _light_ level.
Precondition: port is 1, 2, or 3.

Precondition: The port has been set up as a light sensor.

Precondition: light is between 0 and 100, inclusive.

void waitForDarker (int port, int dif)

Purpose: Makes the program wait until the light sensor reading on port is decreased by dif.

Precondition: port is 1, 2, or 3.

Precondition: The port has been set up as a light sensor.

Precondition: dif is between 0 and 100, inclusive. 0 is no wait at all. Real light levels never really change by anything like 100.

void waitForDark (int port, int light)

Purpose: Makes the program wait until the light sensor reading on port is at most the desired light level.

Precondition: port is 1, 2, or 3.

Precondition: The port has been set up as a light sensor.

Precondition: light is between 0 and 100, inclusive.

int getLight (int port)

Purpose: Returns the current light sensor reading on port.

Precondition: port is 1, 2, or 3.

Precondition: The port has been set up as a light sensor.

6.6 Using rotation sensors

6.6.1 Methods

void waitForRotation (int port, int rotation)

Purpose: Makes the program wait until the counter in the rotation sensor on port has changed by at least the absolute value of rotation.

Precondition: port is 1, 2, or 3.

Precondition: The port has been set up as a rotation sensor.
int getRotation (int port)

*Purpose*: Returns the current rotation sensor reading on port.

*Precondition*: port is 1, 2, or 3.

*Precondition*: The port has been set up as a rotation sensor.

```java
void resetRotation (int port)
```

*Purpose*: Sets the counter in the rotation sensor on port to zero.

*Precondition*: port is 1, 2, or 3.

*Precondition*: The port has been set up as a rotation sensor.

### 6.7 Output port constants

#### 6.7.1 Constants

```java
final int A
```

*Purpose*: Constant to select port A.

```java
final int B
```

*Purpose*: Constant to select port B.

```java
final int C
```

*Purpose*: Constant to select port C.

### 6.8 Using motors

#### 6.8.1 Methods

```java
void motorForward (int port, int power)
```

*Purpose*: Make the motor on port go forwards at the given power.

*Precondition*: port is A, B, or C.

*Precondition*: power is between 0 and 7, inclusive.

```java
void motorBackward (int port, int power)
```

*Purpose*: Make the motor on port go backwards at the given power.

*Precondition*: port is A, B, or C.
void motorStop (int port)

*Purpose:* Stop the motor on port.

*Precondition:* port is A, B, or C.

void motorFloat (int port)

*Purpose:* Float the motor on port.

*Precondition:* port is A, B, or C.

### 6.9 Using lamps

#### 6.9.1 Methods

void lampOn (int port, int power)

*Purpose:* Make the lamp on port go on at the given power.

*Precondition:* port is A, B, or C.

*Precondition:* power is between 0 and 7, inclusive.

void lampOff (int port)

*Purpose:* Turns the lamp on port off.

*Precondition:* port is A, B, or C.

### 6.10 Waiting for fixed times

#### 6.10.1 Methods

void sleep (int ms)

*Purpose:* Makes the program wait for a requested number of ms (miliseconds).

### 6.11 Making sounds

#### 6.11.1 Methods

void systemSound (int i)
Purpose: Play system sound number $i$. The system sounds are as follows.

$$
\begin{array}{ll}
  i & \text{description} \\
  0 & \text{short beep} \\
  1 & \text{double beep} \\
  2 & \text{descending arpeggio} \\
  3 & \text{ascending arpeggio} \\
  4 & \text{long, low beep} \\
  5 & \text{quick ascending arpeggio} \\
\end{array}
$$

Precondition: $0 \leq i \leq 5$.

void playTone (int frequency, int duration)

Purpose: Plays a tone, given its frequency (Hertz) and duration (centiseconds).

Precondition: $31 \leq \text{frequency} \leq 2100$.

Precondition: $0 \leq \text{duration} \leq 256$.

6.12 Using the LCD

6.12.1 Methods

void showNumber (int a)

Purpose: Displays a number in the LCD. Does not require refresh().

Precondition: $0 \leq a \leq 9999$.

void clear ()

Purpose: Clears the LCD, but the effect will not show until refresh() is called.

void refresh ()

Purpose: Refreshes the LCD, causing the changes to be displayed.

void putChar (char c, int i)

Purpose: Puts a character $c$ into the LCD at position $i$.

Precondition: $0 \leq i \leq 4$. 

85
6.13 Using infra-red communications

6.13.1 Methods

void sendByte (int i)

*Purpose:* Send one byte of information to another RCX, by infra-red transmission. The value to send, i, will be truncated if it can’t fit in a byte.

int receiveByte ()

*Purpose:* Waits for and returns a new byte sent by another RCX via infra-red transmission.

6.14 Math

6.14.1 Purpose

The following are some commonly used numeric constants and functions.

6.14.2 Constants

final int MAX_INT

*Purpose:* A constant holding the maximum value an int can have, $2^{31} - 1$.

final int MIN_INT

*Purpose:* A constant holding the minimum value an int can have, $-2^{31}$.

final double PI

*Purpose:* The closest double approximation to $\pi$.

final double E

*Purpose:* The closest double approximation to $e$.

6.14.3 Methods

double abs (double a)

*Purpose:* Returns the absolute value of a.

int abs (int a)

*Purpose:* Returns the absolute value of a.
double ceil (double a)

Purpose: Returns the least double value that is greater than or equal to a and equal to an integer.

double exp (double x)

Purpose: Returns $e^x$, that is Euler’s constant $e$ raised to power $x$.

double floor (double a)

Purpose: Returns the greatest double value that is less than or equal to a and equal to an integer.

double log (double x)

Purpose: Returns the natural logarithm of $x$.

double rint (double x)

Purpose: Returns the closest mathematical integer to $x$.

int round (float a)

Purpose: Returns the closest int to a.

double sqrt (double a)

Purpose: Returns the square root of a.

Precondition: $a \geq 0.0$.

double pow (double a, double b)

Purpose: Returns a raised to the power b, $a^b$.

double sin (double a)

Purpose: Returns the trigonometric sine of a radians.

double cos (double a)

Purpose: Returns the trigonometric cosine of a radians.

double tan (double a)

Purpose: Returns the trigonometric tangent of a radians.

double asin (double a)
Purpose: Returns the trigonometric arc sine of a in radians.

double acos (double a)

Purpose: Returns the trigonometric arc cosine of a in radians.

double atan (double a)

Purpose: Returns the trigonometric arc tangent of a in radians.

double atan2 (double y, double x)

Purpose: Returns the angle \( \theta \) from the conversion of rectangular coordinates \((x, y)\) to polar coordinates \((r, \theta)\).

double max (double a, double b)

Purpose: Returns the greater of a and b.

int max (int a, int b)

Purpose: Returns the greater of a and b.

double min (double a, double b)

Purpose: Returns the lesser of a and b.

int min (int a, int b)

Purpose: Returns the lesser of a and b.

double random ()

Purpose: Returns a random value \( x \) such that \( 0.0 \leq x < 1.0 \).

6.15 Strings

6.15.1 Purpose

The following are methods for working with Strings.

6.15.2 Methods

int length (String s)

Purpose: Returns the length of s.

char charAt (String s, int i)

Purpose: Returns the character at position i in s.
Precondition: $0 \leq i < \text{length}(s)$.

boolean equals (String a, String b)

Purpose: Returns true if and only if a contains the same sequence of characters as in b.

boolean parseBoolean (String s)

Purpose: Returns s converted to a boolean.

int parseInt (String s)

Purpose: Returns s converted to an int.

long parseLong (String s)

Purpose: Returns s converted to a long.

float parseFloat (String s)

Purpose: Returns s converted to a float.

double parseDouble (String s)

Purpose: Returns s converted to a double.

6.16 Threads

6.16.1 Rewrites

void run ()

Purpose: A program may have a second thread of execution. The run() method contains the statements that perform the actions of the second thread.

6.16.2 Methods

void start ()

Purpose: Start the extra thread and call run().
Index

A, 51, 73, 83
abs, 11, 26, 43, 63, 64, 76, 86
acos, 12, 27, 45, 65, 78, 88
animate, 36
args, 25
asin, 12, 27, 45, 65, 77, 87
atan, 12, 27, 45, 65, 78, 88
atan2, 12, 27, 45, 65, 78, 88
B, 51, 73, 83
beep, 59
BLACK, 54
BLUE, 54
buzz, 59
C, 51, 73, 83
ceil, 11, 26, 43, 64, 77, 87
carat, 13, 29, 46, 66, 78, 88
clear, 60, 75, 85
clearRect, 34
close, 21
COLOR, 51
contentHeight, 34
contentWidth, 34
cos, 12, 27, 44, 64, 77, 87
drawArc, 35
drawImage, 36
drawInt, 61
drawLine, 35
drawOval, 35
drawPolygon, 35
drawPolyline, 35
drawRect, 35
drawRoundRect, 35
drawString, 35, 60, 61
fallingBeeps, 59
fillArc, 35
fillOval, 35
fillPolygon, 36
fillRect, 36
fillRoundRect, 36
floor, 11, 26, 44, 64, 77, 87
FLUTE, 59
format, 13, 14, 29, 30, 46, 47
getColor, 54
getDistance, 57
getImage, 36
getLight, 53, 72, 82
getRotation, 57, 73, 83
getSound, 37
getVolume, 55
GREEN, 54
isNextBoolean, 8, 18, 23, 41
isNextDouble, 8, 18, 23, 41
isNextFloat, 9, 19, 23, 41
isNextInt, 8, 18, 22, 41
isNextLine, 9, 19, 23, 41
isNextLong, 8, 18, 22, 41
isNextWord, 9, 19, 23, 41
isPushed, 52, 71, 81
lampOff, 58, 74, 84
lampOn, 58, 74, 84
LCD_COLUMNS, 60
LCD_HEIGHT, 60
LCD_LINES, 60
LCD_WIDTH, 60
length, 13, 29, 46, 66, 78, 88
LIGHT, 70, 80
LIGHT_FLOOD, 50
LIGHT_NOFLOOD, 50
LIGHT_RCX, 50
listenPort1, 69
listenPort2, 69
listenPort3, 69
listenPort4, 69
log, 11, 27, 44, 64, 77, 87
90
loopSound, 37
main, 7, 17, 33, 50, 70, 80
max, 12, 28, 45, 65, 78, 88
MAX_INT, 10, 25, 43, 63, 76, 86
MAX_LONG, 10, 26, 43, 63
min, 13, 28, 45, 65, 78, 88
MIN_INT, 10, 25, 43, 63, 76, 86
MIN_LONG, 10, 26, 43, 63
motorBackward, 58, 73, 83
motorFloat, 58, 74, 84
motorForward, 58, 73, 83
motorStop, 58, 74, 84
mouseX, 38
mouseY, 38

NONE, 54
numArgs, 25

onChange1, 68
onChange2, 68
onChange3, 68
onChange4, 68
onKeyPressed, 39
onKeyReleased, 39
onKeyTyped, 39
onMouseClicked, 38
onMouseEntered, 38
onMouseExited, 38
onMousePressed, 38
onMouseReleased, 38
onPressEnter, 61
onPressEscape, 62
onPressLeft, 62
onPressRight, 62
onReleaseEnter, 61
onReleaseEscape, 62
onReleaseLeft, 62
onReleaseRight, 62
openRead, 20
openWrite, 20

paintWindow, 33
parseBoolean, 14, 30, 47, 66, 79, 89
parseDouble, 15, 30, 47, 66, 79, 89
parseFloat, 15, 30, 47, 66, 79, 89
parseInt, 15, 30, 47, 66, 79, 89
parseLong, 15, 30, 47, 66, 79, 89
pause, 59
PI, 10, 26, 43, 63, 76, 86
PIANO, 59
playNote, 59
playSound, 37
playTone, 59, 75, 85
pow, 12, 27, 44, 64, 77, 87
print, 9, 19, 24, 42
println, 9, 10, 19, 20, 24, 25, 42
PROXIMITY, 51
putChar, 75, 85

random, 13, 28, 45, 65, 78, 88
readBoolean, 7, 17, 21, 40
readDouble, 7, 17, 21, 40
readFloat, 8, 18, 22, 40
readInt, 7, 17, 21, 40
readLine, 8, 18, 22, 40
readLong, 7, 17, 21, 40
readWord, 8, 18, 22, 40
receiveByte, 76, 86
RED, 54
refresh, 75, 85
repaint, 36
resetRotation, 57, 73, 83
resumeDebugger, 16, 31, 49
rint, 11, 27, 44, 64, 77, 87
risingBeeps, 59
ROTATION, 70, 80
ROTATION_RCX, 51
round, 11, 27, 44, 64, 77, 87
run, 89
run1, 67
run2, 67
run3, 67
run4, 67
run5, 67

scroll, 61
sendByte, 76, 86
setColor, 36
setFloodlight, 53, 54
setFrameSize, 33
setFrameTitle, 34
setFrameVisible, 34
setPixel, 60
setUpSensor, 51, 70, 80
showNumber, 75, 85
sin, 12, 27, 44, 64, 77, 87
sleep, 39, 62, 74, 84
SOUND, 51
sqrt, 11, 27, 44, 64, 77, 87
start, 89
start1, 67
start2, 67
start3, 67
start4, 68
start5, 68
stopSound, 37
suspendDebugger, 16, 31, 48
systemSound, 74, 84

tan, 12, 27, 44, 65, 77, 87
TOUCH, 50, 70, 80
twoBeeps, 59

waitForDark, 53, 72, 82
waitForDarker, 53, 72, 82
waitForFar, 56
waitForFurther, 56
waitForLetGo, 52, 71, 81
waitForLight, 52, 71, 81
waitForLighter, 52, 71, 81
waitForLoud, 55
waitForLouder, 55
waitForNear, 56
waitForNearer, 56
waitForPush, 52, 71, 81
waitForQuiet, 55
waitForQuieter, 55
waitForRotation, 57, 72, 82
WHITE, 54

XYLOPHONE, 59